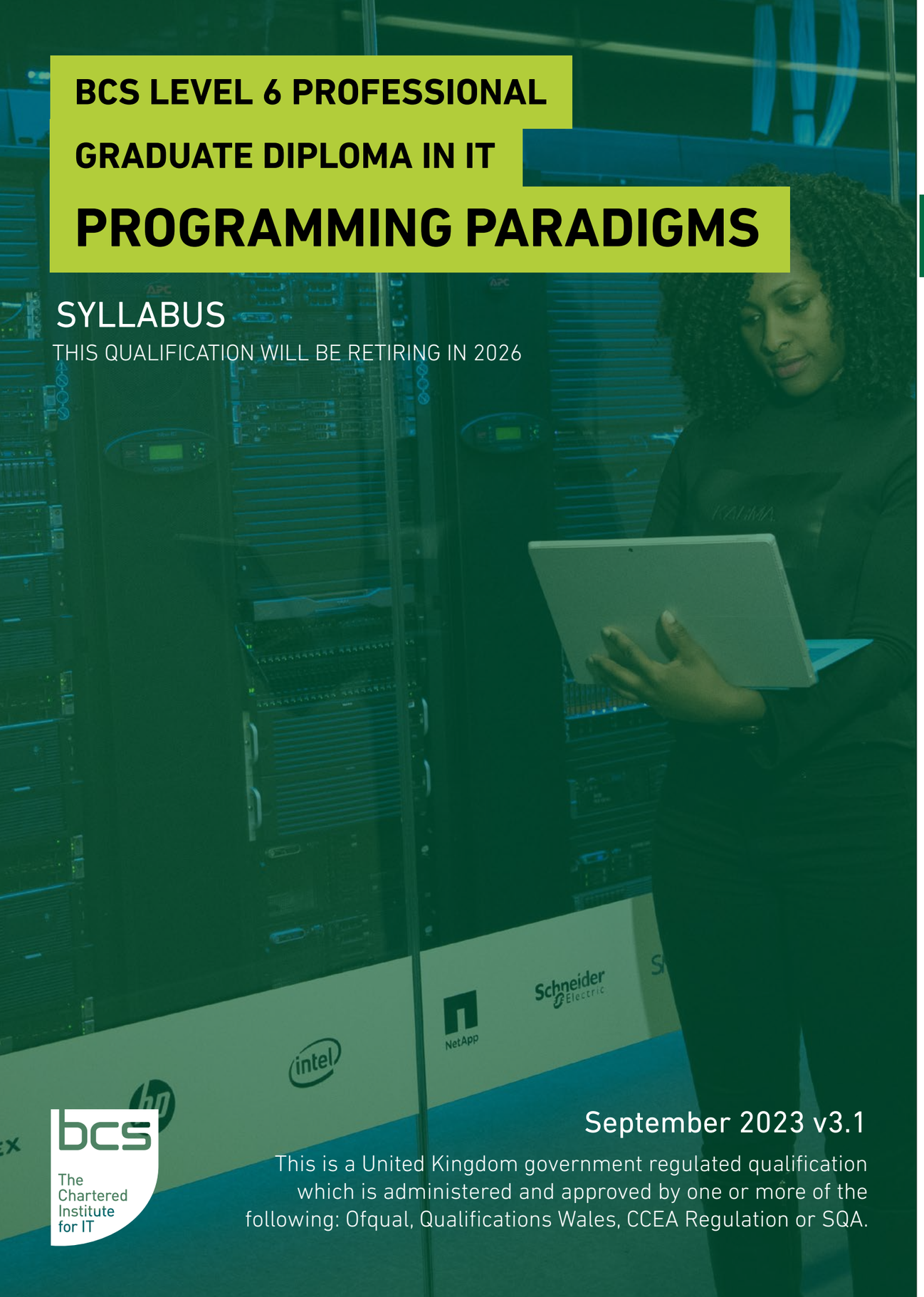


BCS LEVEL 6 PROFESSIONAL GRADUATE DIPLOMA IN IT

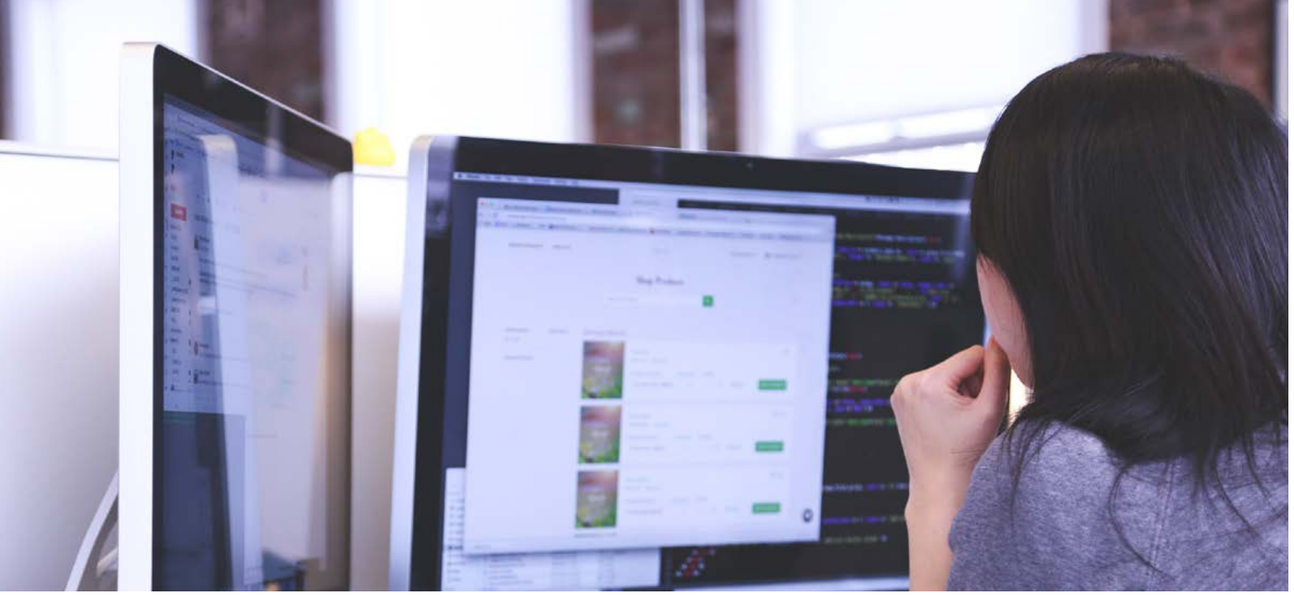
PROGRAMMING PARADIGMS

SYLLABUS

THIS QUALIFICATION WILL BE RETIRING IN 2026

September 2023 v3.1

CONTENTS



Introduction

The final stage within the BCS three-stage Higher Education Qualification programme, the Level 6 Professional Graduate Diploma (PGD) enables candidates who have already achieved the Level 5 Diploma in IT to gain depth of knowledge and expertise in their field.

Our modules have been created in-line with the SFIAPlus framework and latest developments in the industry, giving you a competitive edge in the IT job market and showing your dedication to the industry. You will have the opportunity to learn about topics such as advanced database management, network information systems, web engineering and programming paradigms, as well as to build upon knowledge and skills developed during the Level 5 Diploma.

To successfully achieve the qualification, candidates need to complete:

- One core module (Professional Project in IT)
- Four optional modules

Depending on entrance conditions, completing the Level 6 PGD in IT may support entry onto a Master's degree course at selected global universities.

Programming Paradigms optional module

The Programming Paradigms module is an optional module that forms part of the Level 6 PGD in IT – the final stage within the BCS three-stage Higher Education Qualification programme.

Candidates will develop an overview of modern programming languages and the programming paradigms they implement, as well as an appreciation for the new perspectives on software construction that each language offers. This module explores a range of software development tools and programming techniques, and is designed to showcase the contribution language designers can make to software engineering practice.

Qualification Suitability and Overview

Candidates must have achieved the Diploma in IT or have an appropriate exemption in order to be entered for the Professional Graduate Diploma (PGD). Candidates can study for this PGD by attending a training course provided by a BCS accredited Training Provider or through self-study, although it is strongly recommended that all candidates register with an approved centre. Studying with an approved centre will deliver significant benefits.

Candidates are required to become a member of BCS, The Chartered Institute for IT, to sit and be awarded the qualifications. Candidates may apply for a four-year student membership that will support them throughout their studies.

The Level 6 PGD is suitable for professionals wishing to gain an advanced formal IT qualification, and this module may be particularly relevant for candidates who are interested in career opportunities such as web, app, software or game development.

Total Qualification Time (Certificate)	Guided Learning Hours (Module)	Assessment Time (Exam)
1414 hours	**250 hours**	**Three hours**

SFIA Levels

This award provides candidates with the level of knowledge highlighted within the table, enabling candidates to develop the skills to operate successfully at the levels of responsibility indicated.

Level	Levels of Knowledge	Levels of Skill and Responsibility (SFIA)
K7		Set strategy, inspire and mobilise
K6	Evaluate	Initiate and influence
K5	Synthesise	Ensure and advise
K4	Analyse	Enable
K3	Apply	Apply
K2	Understand	Assist
K1	Remember	Follow

SFIA Plus

This syllabus has been linked to the SFIA knowledge skills and behaviours required at Level 6.

PROG4

Designs, codes, verifies, tests, documents, amends and refactors complex programs/scripts and integration software services. Contributes to selection of the software development approach for projects, selecting appropriately from predictive (plan-driven) approaches or adaptive (iterative/agile) approaches. Applies agreed standards and tools, to achieve well-engineered outcomes. Participates in reviews of own work and leads reviews of colleagues' work..

Further detail around the SFIA Levels can be found at www.bcs.org/levels.

Learning Outcomes

Upon completion of this module, candidates will be able to:

- Compare and contrast a range of programming paradigms.
- Evaluate programming language features critically with respect to the way they support good software engineering practice
- Discuss the appropriateness of the use of a given programming paradigm within a given environment.

Syllabus

1. The nature of programming languages

Learners will be able to:

1.1 Critically compare imperative and declarative languages.

Indicative content	Guidance
a. General characteristics of Imperative languages b. General characteristics of declarative languages	This considers the general distinction between specifying how the computer should perform a task (i.e. an imperative style) and what the computer should do (i.e. a declarative style). Candidates should be able to compare these two general paradigms, consider the application of these for a problem and discuss examples of different languages.

1.2 Discuss different styles of language.

Indicative content	Guidance
a. Object-oriented languages b. Procedural languages c. Scripting languages d. Data-oriented languages	Understanding different styles of programming languages is important to help chose appropriate tools for a problem. Candidates should be aware of the main concepts and the similarities and differences between these language styles. Candidates should also be able to discuss examples of these styles of development and consider when some are more appropriate than others for a given task.

1.3 Discuss event-driven programming and its use.

Indicative content	Guidance
a. Events and event handlers b. Use in programs with graphical user interfaces c. Use in web development	Event-driven programming is typically used in systems with user interfaces, including desktop, mobile and web systems. Candidates should be familiar with the ideas of events and event handlers, and how these are used to develop systems.

1.4 Discuss language standardisation and its use.

Indicative content

a. Purpose of standardisation for a programming language
b. Benefits for developers and tool vendors
c. Potential drawbacks of standardisation
d. Awareness standardisation bodies, e.g. ECMA, ANSI, ISO

Guidance

This section of the syllabus aims to consider the benefits of having a definition of a language that developers and tool vendors can work with and how that helps with portability and tool support. Candidates should be aware of some of the standardisation bodies and be able to discuss possible problems with standardisation in terms of the speed of change and getting agreement for changes.

2. Programming environments

Learners will be able to:

2.1 Describe the use of compilers and interpreters and how they work.

Indicative content

a. Process of converting code into executable programs
b. Compilation
c. Interpreters
d. Hybrid use of compilation and interpreters
e. Steps for compilation and interpreters, e.g.:
   i. Tokenising
   ii. Parsing
   iii. Code generation
   iv. Linking

Guidance

Computer programs need to be converted from some human-readable format into a format that can be executed on the computer. Different languages use compilation, interpretation or a mixture of these techniques to produce executable code. Candidates should be familiar with the concepts and understand that different languages require their own tools for this process. They should have an awareness of the similarities and differences of the main processing stages for compilers and interpreters.

2.2 Discuss interactive development tools (IDE) and their use.

Indicative content

a. Purpose and architecture/ framework of IDE
b. Tools available, e.g.:
   i. Code editing
   ii. Syntax highlighting
   iii. Code completion
   iv. Debugging
   v. Testing
   vi. Build tools
   vii. Version control
h. Support for team development

Guidance

This considers how IDEs provide a way to bring together a set of development tools into a single development environment. There are numerous IDEs available, some of which are designed for a single language or development environment and some that provide support for multiple languages. Candidates should have an awareness of the typical tools that are available and how they can be used to support development of a system and also collaboration within a team.

2.3 Discuss the purpose and use of debugging tools.

Indicative content

a. Purpose of debugging in system development
b. Common features in debugging tools, e.g.:
   i. Run
   ii. Pause
   iii. Step into
   iv. Step over
c. Inspecting values
d. Watch values
e. Breakpoints

Guidance

Debugging tools provide invaluable support to developers to interrogate the state of a system while it is running. Candidates should understand the key functions and how these can be used to investigate and solve problems in the logic of the system.

2.4 Discuss the purpose and use of testing tools.

Indicative content

a. Purpose of testing tools
b. Automated testing, e.g.:
   i. Using unit testing frameworks
   ii. UI testing frameworks
c. Use of static checking, e.g.:
   i. Lint
e. Role of testing in continuous integration

Guidance

Candidates should show an understanding of the relevance of testing tools for software development and how tests can be automated to have repeatable tests for different parts of the system. Automated tests may include testing the code during execution or running static analysis to catch common issues. There should be an awareness of how continuous integration is used in modern systems to run tests.

2.5 Describe configuration management.

Indicative content

a. Build tools
b. Version control systems, e.g.:
   i. Git
   ii. Subversion
c. Building different versions of software

Guidance

Configuration management is used to manage dependencies, build software, track the development of different versions of the software, and rebuild specific versions of software systems. Candidates should be able to discuss the general processes and example tools that can support this.

3. Object orientation

Learners will be able to:

3.1 Discuss concepts of object-oriented programming.

Indicative content

a. Basic concepts, e.g.:
   i. Objects
   ii. Classes
   iii. Methods
   iv. Overloading methods
   v. Messages
b. Inheritance, e.g.:
   i. Single inheritance
   ii. Multiple inheritance
   iii. Overriding methods
   iv. Interfaces (e.g. in Java)
   v. Generalisation
c. Encapsulation
d. Polymorphism

Guidance

Object oriented systems are a popular way of developing modern systems and they are an example of imperative programming. Candidates should be able to describe the main concepts and apply them to example problems to show how the concepts are used in programs.

4. Functional programming

Learners will be able to:

4.1 Discuss the concepts in functional programming.

Indicative content

a. Definition of functions
b. Domain and range of functions
c. Total and partial functions
d. Strict functions
e. Recursive functions and differences between recursion and iteration
f. Examples of applying these concepts to a sample problem

Guidance

Functional programming is a programming paradigm that creates programs by composing and applying functions. As well as functional languages such as Haskell, some of the functional concepts are being adopted in other programming languages. An awareness of these concepts gives candidates an alternative way to think about developing software, and an understanding of how these concepts can be applied in other programming languages. Candidates should be able to write a short extract of code in a functional language, e.g. Haskell.

4.2 Discuss the concept of side-effects and referential transparency.

Indicative content

a. Purpose of side-effects
b. Pure function
c. Referential transparency

Guidance

Candidates should have an understanding of side-effects and their relevance for functional and non-functional languages.

5. Logic programming

Learners will be able to:

5.1 Discuss the concepts in logic programming.

Indicative content

a. Definition and semantics of a logic program
b. Facts, queries and rules, atoms, types and structures
c. Recursion
d. First-order logic

Guidance

This aims to provide a grounding in logic programming so that a learner understands the main concepts and how they compare and contrast with other programming paradigms. Candidates should be familiar with Prolog as an example logic language and be able to understand and write simple programs.

5.2 Discuss the use of queries.

Indicative content	Guidance
a. Existential queries b. Conjunctive queries c. The application of rules	This section of the syllabus is about using queries to interrogate the knowledgebase as part of a logic program. This is an important aspect that makes active use of the facts that are defined in the system.

5.3 Discuss and show understanding of goal reduction.

Indicative content	Guidance
a. Specifying goals b. Control flow, e.g. cut operator	Candidates should be able to demonstrate an understanding of how goals are processed and how the control flow can be managed

5.4 Discuss negation in logic programming.

Indicative content	Guidance
a. Closed world assumption b. Negation as failure	This is about what knowledge is contained within a logic program and what that means for processing when knowledge is not contained within the knowledge base.

6. Related issues

Learners will be able to:

6.1 Discuss the term concurrency.

Indicative content	Guidance
a. The purpose of concurrent systems b. Mutual exclusion blocks c. Semaphores d. Race condition e. Deadlock	Modern computer systems run concurrently, and candidates are expected to be aware of the key issues of concurrency and how it can be used in systems. They should have an awareness of general techniques such as how to handle shared data between the concurrent parts of the system. Candidates should also be able to apply the ideas to a problem, and also show an awareness of problems that can exist in concurrent systems.

Examination Format

This module is assessed through completion of an invigilated written exam.

Type	Three written questions from a choice of five, each with equal marks
Duration	Three hours
Supervised	Yes
Open Book	No (no materials can be taken into the examination room)
Passmark	10/25 (40%)
Delivery	Paper format only

Adjustments and/or additional time can be requested in line with the BCS reasonable adjustments policy for candidates with a disability or other special considerations.

Question Weighting

Candidates will choose three questions from a choice of five. All questions are equally weighted and worth 25 marks.

Recommended Reading

Primary texts

Title: Concepts of Programming Languages
Author: Sebesta, R.
Publisher: Pearson
Date: 2016
ISBN: 978-1292100555

Title: Programming Language Pragmatics
Author: Scott, M.
Publisher: Morgan Kaufmann
Date: 2015
ISBN: 978-8131222560

Title: Programming Languages: Principles and Paradigms
Author: Gabbrielli, M., Martini, S.
Publisher: Springer
Date: 2010
ISBN: 978-1848829138

Additional texts

Title: Programming Language Pragmatics
Author: Scott, M.
Publisher: Morgan Kaufmann
Date: 2015
ISBN: 978-0201710120

Title: Concepts in Programming Languages
Author: J. C. Mitchell
Publisher: Cambridge University Press
Date: 2002
ISBN: 978-0521780988

Title: Programming Languages: Principles and Paradigms (second edition)
Author: A. Tucker and R. Noonan
Publisher: McGraw-Hill
Date: 2006
ISBN: 978-0072866094

Using BCS Books

Accredited Training Organisations may include excerpts from BCS books in the course materials. If you wish to use excerpts from the books you will need a license from BCS. To request a license, please contact the Head of Publishing at BCS outlining the material you wish to copy and its intended use.

Document Change History

Any changes made to the syllabus shall be clearly documented with a change history log. This shall include the latest version number, date of the amendment and changes made. The purpose is to identify quickly what changes have been made.

Version Number	Changes Made
Version 1.0 **July 2021**	Document Creation

CONTACT

For further information please contact:

BCS
The Chartered Institute for IT
3 Newbridge Square
Swindon
SN1 1BY

T +44 (0)1793 417 445

www.bcs.org